Green University of Bangladesh
Department of Computer Science and Engineering(CSE)

Faculty of Sciences and Engineering

Semester: (Fall, Year:2022), B.Sc. in CSE (Day)

Project REPORT

Course Title: Algorithms Lab

Course Code: 206 **Section:** DB

Project Name: The Knight Travails.

Student Details

	Name	ID
1.	A.K.M. Atiqur Rahman	202002035

Submission Date **:** 12-01-2022

Course Teacher's Name **:** Zeseya Sharmin (Ma'am)

[For Teachers use only: Don't Write Anything inside this box]

Lab Report Status

Marks: …… **Signature:.....**

Comments:..... **Date:.....**

Table of Contents

Chapter-1

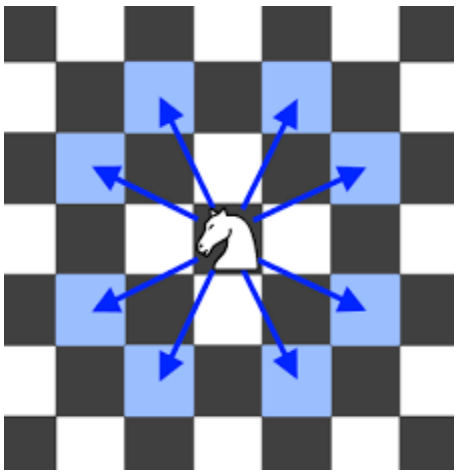
Introduction

1.1 TITLE OF THE PROJECT

The Knight Travails.

1.2 INTRODUCTION

A knight's tour is a series of moves that knight makes while visiting every cell of n x n of the chessboard exactly once. Consider the n x n chessboard as a knight's graph G = (V, E) where vertices Є V represent the cell of the chessboard and every edge Є E represent a knight's move from one cell to another. The knight tour can either be open or close. A knight's tour is a closed tour if a knight starts from one cell and visits all other cells of the n x n chessboard exactly once and the start position is reachable by one knight's move from the last visited cell. Otherwise it is an open path. A knight's close tour is defined to be a Hamiltonian cycle on a knight's graph and an open path is defined to be a Hamiltonian path. Here, approach solving this problem for finding the solution for base cases using the backtracking technique. Distinguished between open path and closed tour once the tour was computed. For a person who is not familiar with chess, the knight valid moves two squares horizontally and one square vertically, or two squares vertically and one square horizontally as shown in the picture given below-



1.3 OBJECTIVES/AIM

By solving this problem, I will learned about -

- ➢ To find combination moves made by a knight and solve general Knight's tour problem.
- ➢ To practice on Backtracking for a problem.
- ➢ To understand how the knight travails moves and knowledge about chess boards

Chapter 2

Design/Development/Implementation of the Project

2.1 TOOLS & TECHNOLOGIES

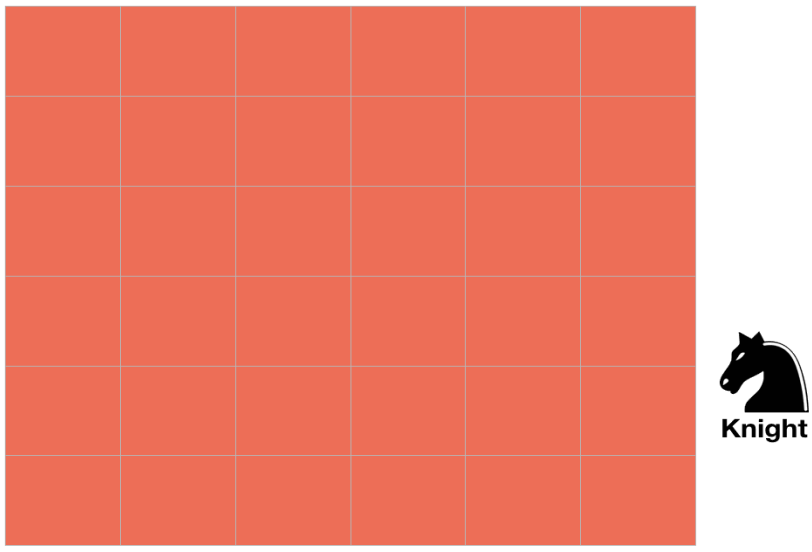
Uses tools and technologies for the whole project -

1. Desktop/Laptop
2. Code Blocks
3. C++ Programming Language

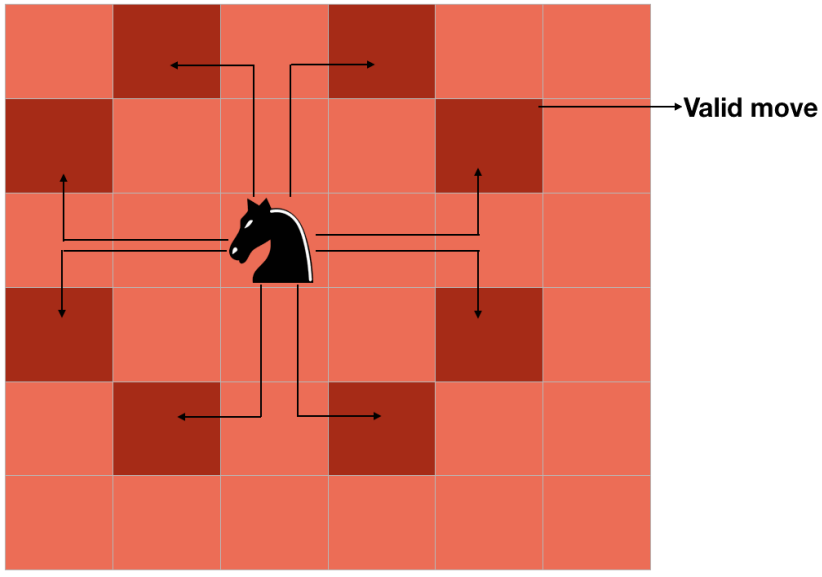
2.2 CONCEPTS USED

1. Algorithms
2. Backtracking
3. Recursion.
4. Object-Oriented Programming.
5. Conditional Statements like while, for, do-while.
6. Chess Concepts.

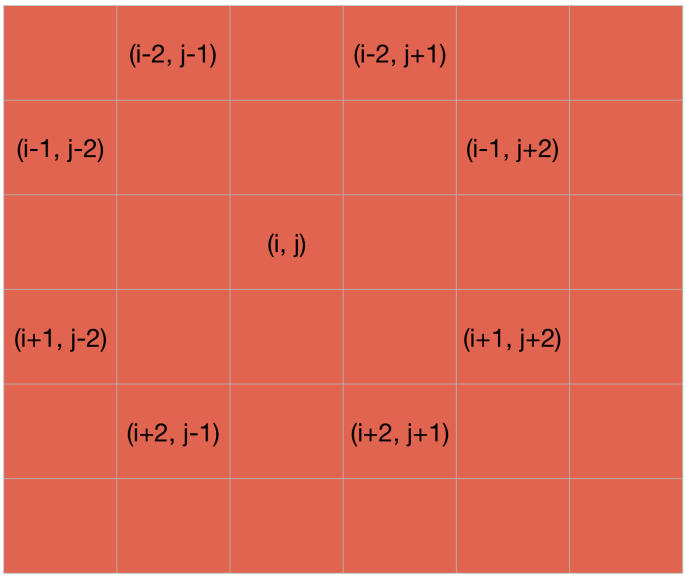
2.3 Diagram



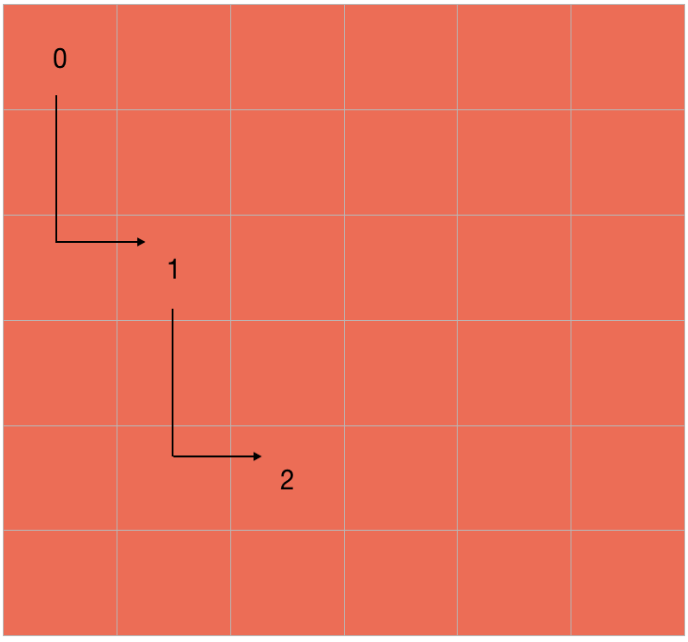
Knight's tour is a problem in which we are provided with a nxn chessboard and knight.



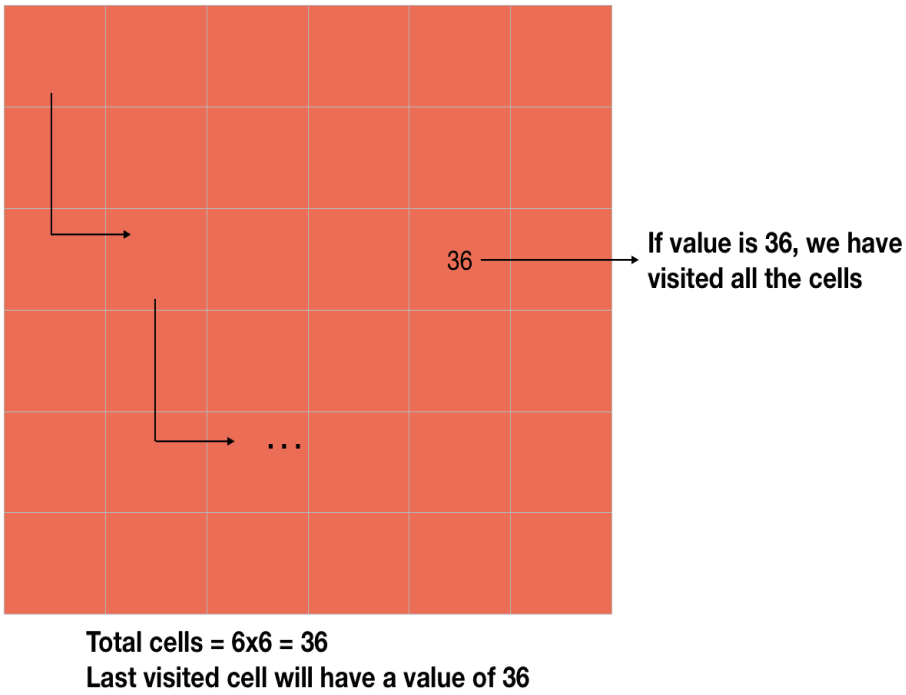
The knight valid moves two squares horizontally and one square vertically, or two squares vertically and one square horizontally



As you can see from the picture above, there is a maximum of 8 different moves which a knight can take from a cell. So if a knight is at the cell (i, j), then the moves it can take are - (i+2, j+1), (i+1, j+2), (i-2,j+1), (i-1, j+2), (i-1, j-2), (i-2, j-1), (i+1, j-2) and (i+2, j-1).



We keep the track of the moves in a matrix. This matrix stores the step number in which we visited a cell. For example, if we visit a cell in the second step, it will have a value of 2.



This matrix also helps us to know whether we have covered all the cells or not. If the last visited cell has a value of N*N, it means we have covered all the cells.

2.4 Algorithm

Input: Dimension of the chessboard and start position.

validbox(x, y, chessboard)

```
if (x>=0 and x<=N and y>=0 and y<=N and chessboard[x][y]==-1)
    return TRUE
return FALSE
```

knight_Travails(chessboard, x, y, move, dx, dy, n)

```
if moves == N*N
    return TRUE
for i in 1 to 8
    nx=x+dx[i]
    ny=y+dy[i]
    if valid_box(nx,ny,n,chessboard)
```

```
        chessboard[nx][ny]=moves
        if knight_Travails(chessboard,nx,ny,mv+1,dx,dy,n)
            return TRUE
        chessboard[nx][ny] = -1
return FALSE
```

2.5 PSEUDOCODE

validbox(x, y, chessboard)

IF x greater equal to 0 and x less equal to N and y greater equal to 0 and y less equal to N and the value of all the unoccupied cells equal to -1

SET ret bool to true

SET found bool to false

knight_Travails(chessboard, x, y, move, dx, dy, n)

IF moves number is equal to N*N box number

SET return bool to true

END IF

FOR evaluating eight possible steps for each box travelled

moving to next step row

moving to next step column

IF valid_box(nx,ny,n,chessboard)

setting the chessboard

IF recursively call knight_Travails

SET return bool to true

The cell (x+dx[i], y+dy[i]) of solution matrix -1

END IF

END FOR LOOP

IF solution doesn't exists

SET return bool to false

2.6 IMPLEMENTATION

Overview: The implementation was completed using C++. The implementation follows the pseudo code.

Details:

Here, implementing a knight travails problems. The implementation was done in 3 parts. The function validbox(x, y, chessboard) to check if a move to the cell (x, y) is valid or not . Then knight_Travails(chessboard, x, y, move, dx, dy, n) function is for doing the knight's valid moves and its travails using backtracking. The last function is the main function which is to set the chess board, take input and print the outputs. So, for implementing KNight travails problem implementation code are given below-

Code:

```
#include<iostream>
using namespace std;

bool validbox(int x,int y,int n,int **chessboard)
{
	if(x>=0&&y>=0&&x<n&&y<n&&chessboard[x][y]==-1)
	return true;
	return false;
}
bool knight_Travails(int **chessboard,int x,int y,int mv,int *dx,int *dy,int n){
	if(mv==n*n)
	{
	return true;
	}
	for(int i=0;i<8;i++)
```

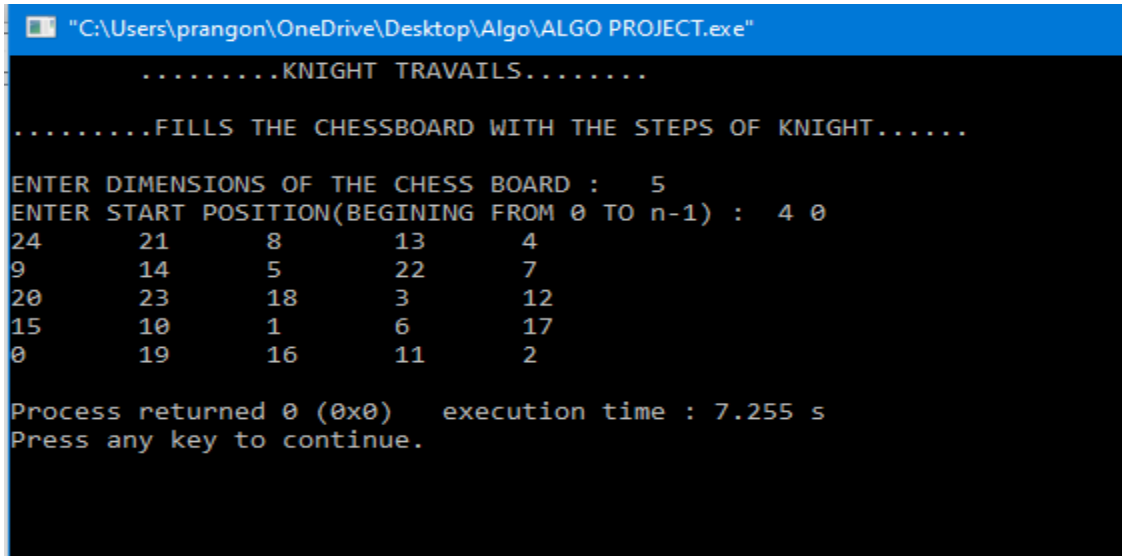
```
        {
        int nx=x+dx[i];
        int ny=y+dy[i];
        if(validbox(nx,ny,n,chessboard))
        {
            chessboard[nx][ny]=mv;
          if(knight_Travails(chessboard,nx,ny,mv+1,dx,dy,n))
              return true;
          chessboard[nx][ny]=-1;
         }
        }
        return false;
}
int main()
{
        int n,x,y;
   cout<<"\t.....KNIGHT TRAVAILS.....\n"<<endl;
   cout<<".....FILLS THE CHESSBOARD WITH THE STEPS OF KNIGHT.....\n"<<endl;
   cout<<"ENTER DIMENSIONS OF THE CHESS BOARD :    ";
   cin>>n;
   int **chessboard=new int*[n];
   for(int i=0;i<n;i++)
        {
        chessboard[i]=new int [n];
```

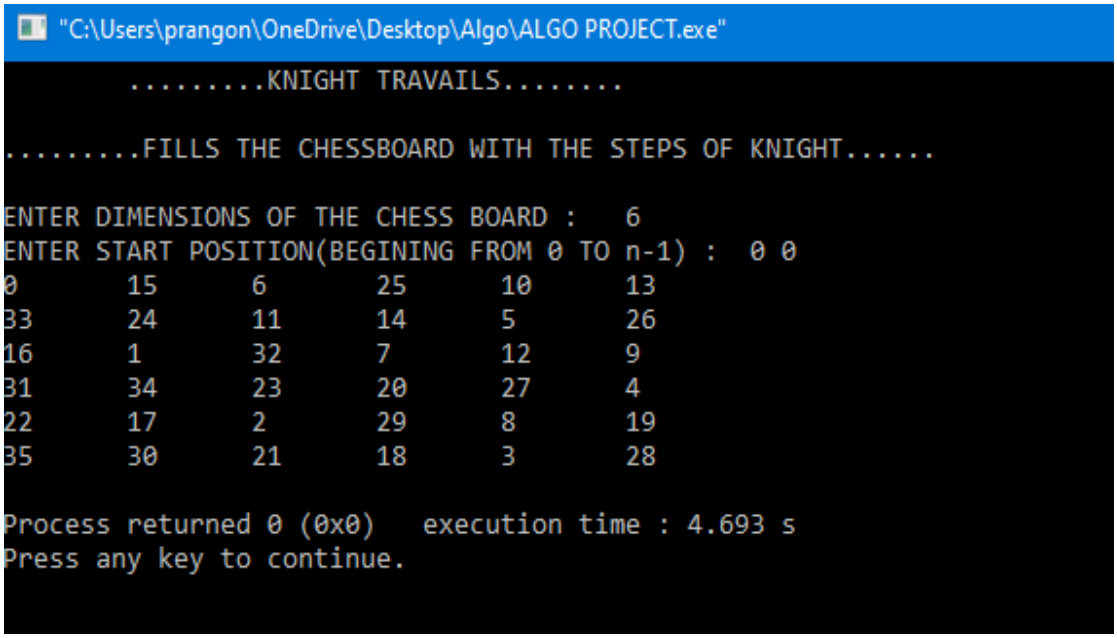
```
        for(int j=0;j<n;j++)
        chessboard[i][j]=-1;
    }

    int dx[]={ 2, 1, -1, -2, -2, -1, 1, 2 };
    int dy[]={ 1, 2, 2, 1, -1, -2, -2, -1 };
    cout<<"ENTER START POSITION(BEGINNING FROM 0 TO n-1) :    ";
    cin>>x>>y;
    chessboard[x][y]=0;
    int mv=1;
    if(knight_Travails(chessboard,x,y,mv,dx,dy,n)){
        for(int i=0;i<n;i++){
        for(int j=0;j<n;j++)
       cout<<chessboard[i][j]<<"    ";
        cout<<"\n";
        }
    }
    else
        cout<<"There is NO SOLUTION AVAILABLE !!";
        return 0;
}
```

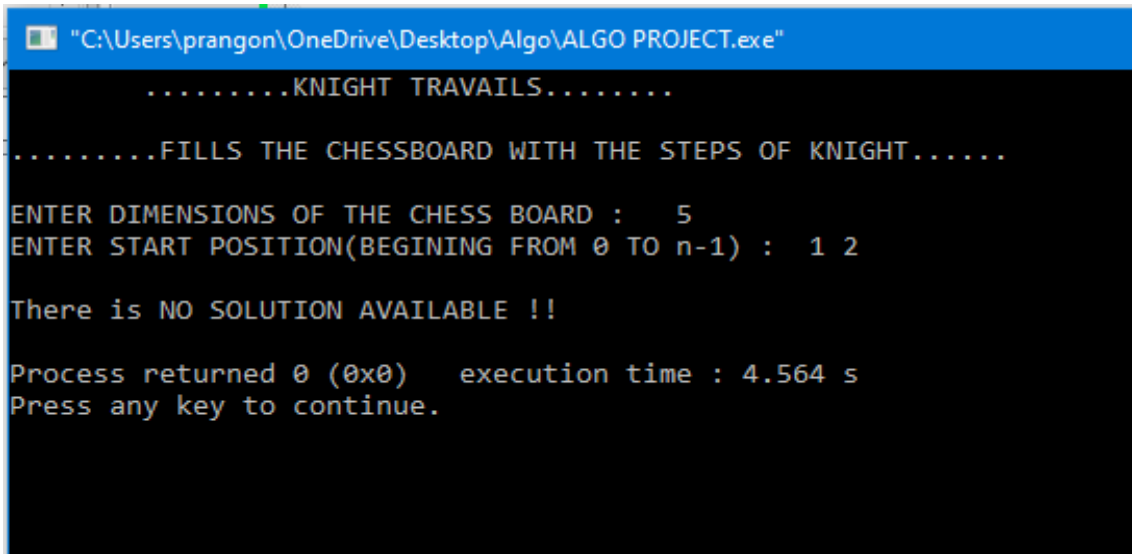
Chapter-3

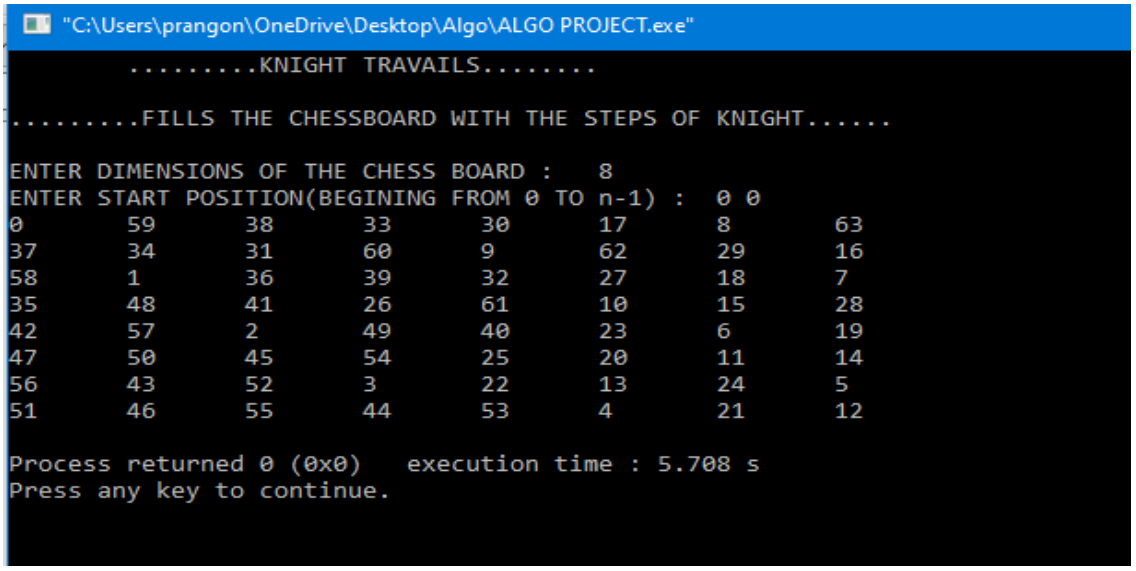
Performance Evaluation

3.1 TEST RESULT / OUTPUT









1. Input : Dimension of the chessboard base case n= 5,6,7,8 and start position.
2. The output will print the Solution and if there exists no solution print no solution.
3. So, that I agree that this program can be used for Knight travel problems.
3. The result is accurately found. So, in this program there are no bugs.

3.2 ACHIEVEMENT

By solving this problem in this project, I achieved all the objectives with the main concept of this project and also with the desired outputs.

3.3 CHALLENGE FACE

Apparently there are almost no bugs in this project. But faced some challenges when doing the tasks which are given below -

- ➢ When selecting the base cases.
- ➢ The problem has some limitations which are that this project has no open path or close cycle for the chessboard with n =3, 4 and a knight's travails for the chessboard of size n x n with base cases n = 5,6,7 and 8 but not for n ≥ 8.

Chapter-4

Conclusion

4.1 SCOPE OF FUTURE WORK

The knight travails project should have scope of future work which are given below -

- ➢ As of now we have implemented a knight's travails for the chessboard of size n x n with base cases n = 5, 6, 7 and 8. • We plan to find a solution for m x n size chessboard .
- ➢ A knight travails problem is an ancient puzzle which remains as a focus of current researchers. This project motivates me to work more on knight all combination moves in different sizes of chess boards and solving this problem by different algorithms takes less amount of time than that of using backtracking.

4.2 DISCUSSION

This project is intended to build a program that solves a Knight's travails in chess boards of size nxn; for any starting nodes that the user provides. There can exist more than one such tours possible in a chess board for provided starting nodes. And this project finds a single solution out of many. By solving this project, I will be able to learn to find combination moves made by a knight and solve general Knight's tour problems and to understand how the knight travails moves and knowledge about chess boards. I will be implementing a knight's travails for the chessboard of size n x n with base cases n = 5, 6, 7 and 8. But in the future I plan to find a solution for m x n size chessboard. This project has no open path or close cycle for the chessboard with n =3, 4.. But, for all values of $n \geq 10$, divide-and-conquer approach takes less amount of time than that of using backtracking. backtracking gave us O(n!) running time. Generally, backtracking is used when we need to check all the possibilities to find a solution and hence it is expensive. For problems like N-Queen and Knight's tour, there are approaches which take less time than backtracking, but for a small size input like 4x4 chessboard, we can ignore the running time and the backtracking leads us to the solution .There are N*N= N^2 cells in the board and we have a maximum of 8 choices to make from a cell, so we can write the worst case running time as $O(8^{N^2})$. Distinguished between open path and closed tour once the tour was computed. Here, also used concepts recursion, chess concepts etc. These projects help me to work on new things in future using these concepts and if ignore the limitation and bugs then successfully done this project.

4.2 REFERENCE

- https://en.wikipedia.org/wiki/Knight's_tour
- https://www.sciencedirect.com/science/article/pii/S0166218X04003488
- https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwivi9uz56r1AhXDT2wGHbWNA_QQFnoECBEQAQ&url=https%3A%2F%2Fwww.geeksforgeeks.org%2Fthe-knights-tour-problem-backtracking-1%2F&usg=AOvVaw0IkCHH7N-jzctAXHibicSf